Reverse Engineering Report – Crackme Challenge

Author: **Ali Rashchi** Github: **AliMMA17** Date: **2025-08-21**

Contact: **alirashchi7@gmail.com**

Legal & Ethical Considerations

This work was performed exclusively for educational purposes on a publicly available crackme challenge designed for reverse engineering practice. The patched binary will not be redistributed. All modifications and analysis respect the platform's guidelines.

Challenge information

Challenge: https://crackmes.one/crackme/5fe5ed8633c5d4264e5900d9

Title: **aolvos's Jeez**

Platform: **Windows (x86)**

Difficulty: **2.0**

Language: **C/C++**

Objective

FIND OUT HOW SERIAL KEYS ARE VALIDATED FROM JEEZ.EXE AND MAKE DESCRIPTION OF THIS ALGORITHM..

Tools Used

- x32dbg
- CFF Explorer
- Python file (for keygen implementation)

Analysis

```
00401A8F    90              nop
00401A90 $  55              push ebp                    sub_401A90
00401A91 .  57              push edi                    edi:sub_401DD0
00401A92 .  56              push esi
00401A93 .  53              push ebx
00401A94 .  81EC 8C000000   sub esp,8C
00401A9A .  A1 34604000     mov eax,dword ptr ds:[406034]
00401A9F .  8D5C24 4F       lea ebx,dword ptr ss:[esp+4F]
00401AA3 .  C74424 08 08000000 mov dword ptr ss:[esp+8],8   int nMaxCount = 08
00401AAB .  895C24 04       mov dword ptr ss:[esp+4],ebx    LPTSTR lpString
00401AAF .  890424          mov dword ptr ss:[esp],eax      HWND hWnd
00401AB2 .  E8 790A0000     call <JMP.&GetWindowTextA>      GetWindowTextA
00401AB7 .  83EC 0C         sub esp,C
00401ABA .  891C24          mov dword ptr ss:[esp],ebx      const char* str
00401ABD .  C74424 58 00000000 mov dword ptr ss:[esp+58],0
00401AC5 .  C74424 5C 00000000 mov dword ptr ss:[esp+5C],0
00401ACD .  C74424 60 00000000 mov dword ptr ss:[esp+60],0
00401AD5 .  C74424 64 00000000 mov dword ptr ss:[esp+64],0  [esp+64]:LoadCursorFromFileW+439
00401ADD .  E8 16130000     call <JMP.&strlen>              strlen
00401AE2 .  83F8 03         cmp eax,3
00401AE5 .  89C6            mov esi,eax
00401AE7 .  C64424 2D 00    mov byte ptr ss:[esp+2D],0
00401AEC .  7F 12           jg jeez.401B00
```

Get the username from the windowsText and check if it's at least 3 characters or not

```
00401B51 .  83C0 05         add eax,5
00401B54 .  39D0            cmp eax,edx
00401B56 .^ 75 DF           jne jeez.401B37
00401B58 .  8D4424 3D       lea eax,dword ptr ss:[esp+3D]
00401B5C .  897C24 04       mov dword ptr ss:[esp+4],edi    const char* strSource = edi:sub_401DD0
00401B60 .  C74424 08 06000000 mov dword ptr ss:[esp+8],6   size_t count = 06
00401B68 .  8D7C24 38       lea edi,dword ptr ss:[esp+38]   edi:sub_401DD0
00401B6C .  8D6E FD         lea ebp,dword ptr ds:[esi-3]
00401B6F .  890424          mov dword ptr ss:[esp],eax      char* strDest
00401B72 .  E8 89120000     call <JMP.&strncpy>             strncpy
00401B77 .  8D4424 71       lea eax,dword ptr ss:[esp+71]
00401B7B .  C74424 08 04000000 mov dword ptr ss:[esp+8],4   size_t count = 04
00401B83 .  C64424 43 00    mov byte ptr ss:[esp+43],0
00401B88 .  894424 04       mov dword ptr ss:[esp+4],eax    const char* strSource
00401B8C .  8D4424 2E       lea eax,dword ptr ss:[esp+2E]
00401B90 .  890424          mov dword ptr ss:[esp],eax      char* strDest
00401B93 .  E8 68120000     call <JMP.&strncpy>             strncpy
00401B98 .  8D4424 76       lea eax,dword ptr ss:[esp+76]
00401B9C .  C74424 08 04000000 mov dword ptr ss:[esp+8],4   size_t count = 04
00401BA4 .  C64424 32 00    mov byte ptr ss:[esp+32],0
00401BA9 .  894424 04       mov dword ptr ss:[esp+4],eax    const char* strSource
00401BAD .  8D4424 33       lea eax,dword ptr ss:[esp+33]
00401BB1 .  890424          mov dword ptr ss:[esp],eax      char* strDest
00401BB4 .  E8 47120000     call <JMP.&strncpy>             strncpy
00401BB9 .  8D4424 7B       lea eax,dword ptr ss:[esp+7B]
00401BBD .  C74424 08 04000000 mov dword ptr ss:[esp+8],4   size_t count = 04
00401BC5 .  893C24          mov dword ptr ss:[esp],edi      char* strDest = edi:sub_401DD0
00401BC8 .  C64424 37 00    mov byte ptr ss:[esp+37],0
00401BCD .  894424 04       mov dword ptr ss:[esp+4],eax    const char* strSource
00401BD1 .  E8 2A120000     call <JMP.&strncpy>             strncpy
```

Copying characters part by part to the stack (first 6, then 4,4,4)

```
00401BD6 .  C64424 3C 00    mov byte ptr ss:[esp+3C],0
00401BDB .  31C9            xor ecx,ecx                 ecx:sub_401DD0
00401BDD .  31D2            xor edx,edx
00401BDF .  90              nop
00401BE0 >  0FBE0413        movsx eax,byte ptr ds:[ebx+edx]
00401BE4 .  83C2 01         add edx,1
00401BE7 .  01C1            add ecx,eax                 ecx:sub_401DD0
00401BE9 .  39EA            cmp edx,ebp
00401BEB .^ 75 F3           jne jeez.401BE0
```

EBX is the address to the first character of username, and ebp = len string – 3 so this part of the code , sum the bytes of the username string (not including the last 3)

```
00401BED  .  0FBE4434 4C      movsx eax,byte ptr ss:[esp+esi+4C]
00401BF2  .  8D6C24 44        lea ebp,dword ptr ss:[esp+44]
00401BF6  .  894C24 58        mov dword ptr ss:[esp+58],ecx
00401BFA  .  BB 01000000      mov ebx,1
00401BFF  .  894424 5C        mov dword ptr ss:[esp+5C],eax
00401C03  .  0FBE4434 4D      movsx eax,byte ptr ss:[esp+esi+4D]
```

Save the last 3 username characters username[len-3], username[len-2], username[len-1] , we call it U0,U1 , U2

```
00401C11  .  8D7424 28        lea esi,dword ptr ss:[esp+28]
00401C15  .  897C24 04        mov dword ptr ss:[esp+4],edi          const char* strSource
00401C19  .  C74424 08 02000000 mov dword ptr ss:[esp+8],2          size_t count = 02
00401C21  .  31FF             xor edi,edi                           edi:sub_401DD0
00401C23  .  893424           mov dword ptr ss:[esp],esi            char* strDest
00401C26  .  894424 64        mov dword ptr ss:[esp+64],eax
00401C2A  .  E8 D1110000      call <JMP.&strncpy>                   strncpy
00401C2F  .  896C24 08        mov dword ptr ss:[esp+8],ebp
00401C33  .  C74424 04 4F404000 mov dword ptr ss:[esp+4],jeez.40404F const char* format
00401C3B  .  893424           mov dword ptr ss:[esp],esi            const char* buffer
00401C3E  .  E8 C5110000      call <JMP.&sscanf>                    sscanf
00401C43  .  8D4424 3A        lea eax,dword ptr ss:[esp+3A]
00401C47  .  C74424 08 02000000 mov dword ptr ss:[esp+8],2          size_t count = 02
00401C4F  .  893424           mov dword ptr ss:[esp],esi            char* strDest
00401C52  .  894424 04        mov dword ptr ss:[esp+4],eax          const char* strSource
00401C56  .  E8 A5110000      call <JMP.&strncpy>                   strncpy
00401C5B  .  8D4424 48        lea eax,dword ptr ss:[esp+48]
00401C5F  .  C74424 04 4F404000 mov dword ptr ss:[esp+4],jeez.40404F const char* format
00401C67  .  893424           mov dword ptr ss:[esp],esi            const char* buffer
00401C6A  .  894424 08        mov dword ptr ss:[esp+8],eax
00401C6E  .  E8 95110000      call <JMP.&sscanf>                    sscanf
```

Save Hex number of D[0] D[1] in X and save hex number of D[2][D[3] in Y ,

[esp+44] = X, [esp+48] = Y

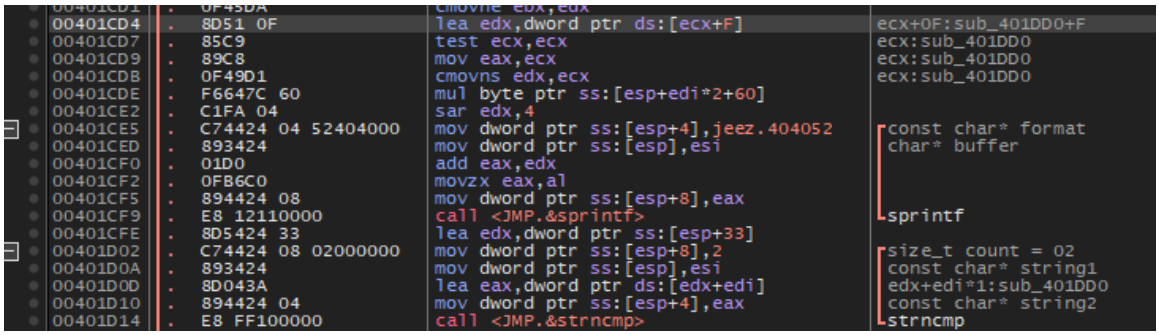
```
00401C73  .  896C24 1C        mov dword ptr ss:[esp+1C],ebp
00401C77  >  8B4424 1C        mov eax,dword ptr ss:[esp+1C]
00401C7B  .  8B50 04          mov edx,dword ptr ds:[eax+4]
00401C7E  .  8D42 0F          lea eax,dword ptr ds:[edx+F]
00401C81  .  85D2             test edx,edx
00401C83  .  0F48D0           cmovs edx,eax
00401C86  .  0FB6447C 58      movzx eax,byte ptr ss:[esp+edi*2+58]
00401C8B  .  F6647D 00        mul byte ptr ss:[ebp+edi*2]
00401C8F  .  C1FA 04          sar edx,4
```

Load EAX = [esp+1C] = &X, and EDX: [eax+4] = [&X + 4] = [esp+48] = Y and

then EAX = Y + 0xF , If EDX (i.e., Y) were negative, copy EAX (Y+0xF) back into EDX. And B[i] = (BASE * M - (Y>>4)) & 0xFF

```
00401C92  .  C74424 04 52404000 mov dword ptr ss:[esp+4],jeez.404052 const char* format
00401C9A  .  893424           mov dword ptr ss:[esp],esi            char* buffer
00401C9D  .  29D0             sub eax,edx
00401C9F  .  0FB6C0           movzx eax,al
00401CA2  .  894424 08        mov dword ptr ss:[esp+8],eax
00401CA6  .  E8 65110000      call <JMP.&sprintf>                   sprintf
00401CAB  .  8D5424 2E        lea edx,dword ptr ss:[esp+2E]
00401CAF  .  C74424 08 02000000 mov dword ptr ss:[esp+8],2          size_t count = 02
00401CB7  .  893424           mov dword ptr ss:[esp],esi            const char* string1
00401CBA  .  8D043A           lea eax,dword ptr ds:[edx+edi]        edx+edi*1:sub_401DD
00401CBD  .  894424 04        mov dword ptr ss:[esp+4],eax          const char* string2
00401CC1  .  E8 52110000      call <JMP.&strncmp>                   strncmp
00401CC6  .  8B4C7D 00        mov ecx,dword ptr ss:[ebp+edi*2]      ecx:sub_401DD0
```

value = (BASE*M - (Y>>4)) & 0xFF, convert to two hex chars at ESI using "%02X"



C[i] = (M*U + (M>>4)) & 0xFF

Methodology

1. Loaded the crackme in x32dbg and monitored execution after entering username/Serial.
2. Understand the keygen algorithm checker.
3. Wrote python code to generate serial based on the string